[METHOD AND APPARATUS FOR EFFICIENT UTILIZATION OF ELECTRONIC FUSE SOURCE CONNECTIONS]

DESCRIPTION

Background of Invention

[Para 1]   Field of the Invention

[Para 2]   The invention generally relates to the placement of fuses in a semiconductor floorplan. In particular, the invention provides a method for and an apparatus in which the FSOURCE connection is split into multiple nets, allowing more efficient placement of primary fuses in the floorplan and reliable programming thereof. .

[Para 3]   Background of the Invention

[Para 4]   The advent of electronic fuses for connecting or disabling redundant structures in, for example, RAMs has introduced many new problems for the physical design of integrated circuits that use redundant structures to improve manufacturing yield. One such problem is the electrical constraints on the FSOURCE net. FSOURCE ports are the chip interface ports (e.g. wirebond pads, C4 pads, or the like) that supply the voltage necessary to blow each fuse in a fuse macro. While C4 pads are preferred and the terminology "C4 pads" is used hereinafter, it should be understood that any other interface port structure (such as wirebond pads) may be used. For example, in some integrated circuits, a fuse domain includes the following elements:

[Para 5]   1. a single fuse controller to control all the fuse macros

[Para 6]   2. up to 64 primary fuse macros for wafer level fuse blow

[Para 7]   3. 4 secondary fuse macros for module level fuse blow

[Para 8]   4. 2 tertiary fuse macros

[Para 9]   5. a single FSOURCE fatwire IO that supplies current from a chip signal IO to all the fuse macros during fuse blow.

[Para 10]   The wire associated with the FSOURCE is referred to as a "fatwire" because the type of wire used in an FSOURCE connection which has a width that can be over 100 times that of a minimum width wire in the circuit in order to carry sufficient current while limiting voltage drop per unit length during programming of fuses.

[Para 11]   The typical arrangement of a fuse domain is depicted schematically in Figure 1 which shows a single fuse controller connected to primary, secondary and tertiary fuse macros which may contain up to 63, 3 and 1 additional macros each, giving a total of 64, 4 and 2 macros each, respectively. As can be seen in the figure, the primary, secondary and tertiary fuse macros are connected to a single FSOURCE fatwire IO cell, which is in turn connected to an FSOURCE C4 pad. In general, an electronic fuse domain will contain a set of primary fuses, an FSOURCE power connection, and other overhead.

[Para 12]   During manufacturing tests, only one fuse macro in a domain is blown at a time. In order to ensure sufficient voltage to blow a fuse inside a fuse macro, the point to point resistance between the fuse macro and the FSOURCE fatwire IO must be considered. For example, in some systems, this point to point resistance must be held to about 10 ohms. In order to meet this resistance constraint, even considering the cross sectional dimensions of the fatwire, the fuse macro must be located relatively close to the FSOURCE fatwire IO. For a full domain of 70 fuse macros (64 primary + 4 secondary + 2 tertiary), this means that all the fuse macros must cluster near the FSOURCE fatwire IO. This requirement can cause several floorplanning problems during physical layout of the chip because such a cluster of fuse domain blocks takes up a relatively large area on the chip, typically about 2mm x 2mm of surface area. Associated problems include:

[Para 13]   1. Fuses and their placement are not critical in terms of performance of the chip, yet they must cluster as a large composite block. This cluster can perturb the placement of more critical blocks resulting in a less than ideal floorplan.

[Para 14]   2. This large cluster can block the area under signal C4 pads and leave no room for IO cells, for example, for loading signals on or off. Placing an IO cell too far away from its C4 pad will violate the resistance constraint on the connection between the IO cell and its C4 pad. This can render the signal C4 pad unusable. Thus, it is desirable to spread out the fuse macros instead of clustering them.

[Para 15]   Figure 2 illustrates typical placement of a fuse domain that clusters around an FSOURCE fatwire IO cell. As can be seen, the cluster of fuses is so dense that IO cells cannot be placed near the C4 pads on top of this cluster and must be located at a distance. Because the distance from the C4 pads to any IO cell is so large in this arrangement, the resistance through a wire connecting a C4 pad to an IO cell is too great to make the C4 pad usable.

[Para 16]   Figure 3 further and more generally illustrates this problem. In Figure 3, a large placement obstruction (for example, a fixed block or a group of fixed blocks such as a group of IO cells) is located at a position where it would be preferable to locate primary fuse macros. Since the fuse macro once programmed has no effect on the functionality of the chip, in theory its placement away from the other fuses and on the other side of the placement obstruction, as shown, is allowable. However, placement of the primary fuse macro at this location makes the distance from the fuse macro to the FSOURCE fatwire IO too great to meet voltage drop requirements (i.e. the distance is too large or "excessively distal" to meet the $V = IR$ drop adequate for reliable fuse programming, or, more generally, to reliably perform any other functions, for example, due to noise, signal propagation, time, etc.). As a result the fuses inside an excessively distal fuse macro may fail to be blown during testing.

[Para 17]   One potential solution to the problem illustrated in Figure 3 is to split the fuse domain into multiple domains as shown in Figure 4. For

example, if a single domain is split into two domains, instead of dealing with one large cluster of fuse macros that must be placed very close together around one FSOURCE fatwire IO, two smaller clusters of fuse macros may be placed around two FSOURCE fatwire IOs. These smaller clusters can be placed independently and may be spaced further apart within a cluster. Figure 4 illustrates this possibility, showing two separate fuse domains located on opposite sides of a placement obstruction. While this solution alleviates some of the floorplanning problems caused by utilizing a single large cluster, this type of design also has some drawbacks. For example:

[Para 18] 1. In addition to the primary fuse macros, each domain requires one fuse controller, one FSOURCE fatwire IO, 4 secondary fuse macros, and 2 tertiary fuse macros. Therefore, splitting a full domain of 64 primary fuse macros into two domains of 32 primary fuse macros (or other allocations) adds all the overhead of a fuse domain, i.e. a second fuse controller, a second FSOURCE fatwire IO cell, 4 additional secondary fuse macros, and 2 additional tertiary fuse macros. (In Figure 4, this overhead is simplified and depicted as a fuse controller, an FSOURCE fatwire IO cell, a single primary fuse macro, a single secondary fuse macro, and a single tertiary fuse macro.)

[Para 19] 2. Fuse domains may not share fuse macros. If one domain uses all of its fuses and the other domain is using none, the domain that is fully utilized cannot use any of the macros from the empty domain if it should require more fuses. This scenario could lead to discarding product that might otherwise have been repaired, since defects on a chip tend to cluster in regions.

[Para 20] The prior art has thus far failed to provide a solution to the problems associated with efficient placement of fuse macros on the floorplan of a semiconductor chip.

Summary of Invention

[Para 21]   The present invention provides a solution for the floorplanning problems associated with placement of fuses on a semiconductor chip. According to the practice of the invention, instead of breaking up an entire fuse domain into multiple domains in order to provide flexibility of floorplanning, the FSOURCE connection associated with the fuse domain is split into multiple nets. In other words, in the practice of the present invention, a fuse domain may contain a plurality of FSOURCE fatwire IO cells, and a primary fuse macro within a domain may be connected to any of the FSOURCE fatwire IO cells in the domain. Therefore, a fuse macro (primary, secondary, or tertiary) may be located at any distance from the fuse controller, and provision must be made only for the wire that connect the distal fuse macro to the fuse controller (plus the additional FSOURCE IO cell and C4 pad, both of which are co-located with the distal primary fuse macro). Fuse macros in a fuse domain thus are relatively mobile and may be flexibly located on a microchip without increasing the number of fuse controllers and attendant overhead. As illustrated in Figure 5, only the wire connecting the fuse controller to the distal primary fuse macro traverses the area occupied by a placement obstruction, and the distal primary fuse macro and its supplemental FSOURCE IO cell and C4 pad are placed well outside the placement obstruction.

[Para 22]   It is an object of this invention to provide a method for increasing the flexibility of placement of primary fuse macros on a microchip. The method includes the steps of: providing components of a fuse domain (the components including one or more primary fuse macros, overhead associated with the one or more primary fuse macros, and one or more primary FSOURCE interface port structures); arranging the components on the microchip such that at least one of the one or more primary fuse macros is located excessively distal from the one or more primary FSOURCE interface port structures; and providing at least one supplemental FSOURCE interface port structure. The at least one supplemental FSOURCE interface port structure is located proximate to and is operatively connected to the one or more excessively distal primary fuse macros.

[Para 23]   In one embodiment of the method, the overhead comprises a fuse controller, one or more secondary fuse macros, and one or more tertiary fuse macros. The one or more primary FSOURCE interface port structures and the at least one supplemental FSOURCE interface port structure may be a C4 pad, or, alternatively, a wire bond pad. According to one embodiment of the method, the fuse domain components comprise 64 primary fuse macros, 4 secondary fuse macros, and 2 tertiary fuse macros. Further, the at least one supplemental FSOURCE interface port structure may be operatively connected to a plurality of excessively distal primary fuse macros.

[Para 24]   The invention further provides a microchip which includes: components of a fuse domain (the components comprising one or more primary fuse macros, overhead associated with said one or more primary fuse macros, and one or more primary FSOURCE interface port structures; and at least one of the one or more primary fuse macros is located excessively distal from the one or more primary FSOURCE interface port structures); and at least one supplemental FSOURCE interface port structure that is located proximate to and is operatively connected to the one or more excessively distal primary fuse macros.

[Para 25]   In one embodiment of the invention, the overhead of the microchip comprises a fuse controller, one or more secondary fuse macros, and one or more tertiary fuse macros. The one or more primary FSOURCE interface port structures and the at least one supplemental FSOURCE interface port structure may be a C4 pad, or, alternatively, a wire bond pad. According to one embodiment of the method, the fuse domain components comprise 64 primary fuse macros, 4 secondary fuse macros, and 2 tertiary fuse macros. Further, the at least one supplemental FSOURCE interface port structure may be operatively connected to a plurality of excessively distal primary fuse macros.

[Para 26]   The invention further provides a method for blowing fuses within a fuse domain in parallel. The method comprises the steps of: providing components of a fuse domain, (the components including one or more primary

fuse macros, overhead associated with the one or more primary fuse macros, and one or more primary FSOURCE interface port structures); arranging the components on the microchip such that at least one of the one or more primary fuse macros is located excessively distal from the one or more primary FSOURCE interface port structures; providing at least one supplemental FSOURCE interface port structure (the at least one supplemental FSOURCE interface port structure is located proximate to and is operatively connected to the one or more excessively distal primary fuse macros); and simultaneously blowing at least two fuses of the fuse domain, wherein each of the at least two fuses is operatively connected to a different FSOURCE interface port structure within the domain.

[Para 27] In one embodiment of the method, the overhead comprises a fuse controller, one or more secondary fuse macros, and one or more tertiary fuse macros. The one or more primary FSOURCE interface port structures and the at least one supplemental FSOURCE interface port structure may be a C4 pad, or, alternatively, a wire bond pad. According to one embodiment of the method, the fuse domain components comprise 64 primary fuse macros, 4 secondary fuse macros, and 2 tertiary fuse macros. Further, the at least one supplemental FSOURCE interface port structure may be operatively connected to a plurality of excessively distal primary fuse macros.

[Para 28] The present invention further provides an integrated circuit. The integrated circuit comprises: a domain of functional elements (including a circuit to said functional elements and a first off–chip connection for said domain of functional elements, and in which at least one of the functional elements is excessively distal from the first off–chip connection); and a second off–chip connection proximal to the at least one functional element which is excessively distal from the first off–chip connection. In one embodiment of the invention, the functional elements of the integrated circuit are primary fuse macros. In another embodiment, the first and second off–chip connections are FSOURCE interface port structures (e.g. C4 pads or wire bond pads).

Brief Description of Drawings

[Para 29]   Figure 1. Schematic depiction of a domain.

[Para 30]   Figure 2. Schematic representation of the typical placement of a fuse domain that clusters around an FSOURCE fatwire IO cell.

[Para 31]   Figure 3 illustrates that case where a large placement obstruction is located at a position where it would be preferable to locate primary fuse macro.

[Para 32]   Figure 4 illustrates two separate fuse domains located on opposite sides of a placement obstruction.

[Para 33]   Figure 5 illustrates incorporation of a second FSOURCE IO fatwire cell into a fuse domain design.

[Para 34]   Figure 6 illustrates the general algorithm for software used in chip floorplanning.

Detailed Description

[Para 35]   The present invention provides a solution for the floorplanning problems associated with placement of fuses on a semiconductor chip. According to the practice of the invention, instead of breaking up the entire fuse domain into multiple domains, only the FSOURCE connection is split into multiple nets. In other words, a fuse domain may contain multiple FSOURCE fatwire IOs, and a fuse macro within a domain may be connected to any of the FSOURCE fatwire IOs in the domain. However, the fuse macros may not cross domains to connect to an FSOURCE fatwire IO in another domain and only one fuse controller is required.

[Para 36]   In the present context, it is recognized that each FSOURCE fatwire IO cell has an associated C4 pad. The terms FSOURCE fatwire IO, FSOURCE IO

fatwire cell, FSOURCE IO, and FSOURCE C4 are used interchangeably and are meant to include both an FSOURCE fatwire IO cell and its associated C4 pad.

[Para 37]   The present invention allows the desired granularity of fuse macro placement without adding the overhead of creating multiple fuse domains (another fuse macro, 4 secondary fuse macros, 2 tertiary fuse macros, and fuse controller). The only extra overhead is the additional FSOURCE IOs and their associated C4 pads, which are also required in the arrangements of Figures 3 and 4.

[Para 38]   Figure 5 is a schematic depiction of one embodiment of the present invention, in which the minimal overhead required to solve a fuse placement problem is illustrated. As can be seen in Figure 5, one primary fuse macro is located at a distance from the other fuse macros in the domain due to a placement obstruction. A similar scenario was depicted in Figure 3; however, in Figure 3, the distance between the distal fuse macro and the FSOURCE IO fatwire cell was too great to allow the distal fuse to be reliably utilized. Problems associated with routing a fatwire are crowding on the chip surface and the inability to blow a fuse located at a distance. In contrast, in the practice of the present invention, the distal primary fuse macro may be utilized because a second, supplemental FSOURCE IO fatwire cell is incorporated into the fuse domain design. The supplemental FSOURCE IO fatwire cell is proximate to (e.g. within a distance that suitably limits the $v = IR$ voltage drop so that it is not "excessively distal") and operatively connected to the distal primary fuse macro. As used herein, "operatively connected" means that the FSOURCE IO fatwire cell is connected to the distal primary fuse macro in a manner that allows the FSOURCE IO fatwire cell to carry out its normal function, i.e. to provide current to blow the primary fuse macro. As can be seen, the only additional overhead required to permit movement of the primary fuse macro to the distal site is the second FSOURCE IO fatwire cell and the associated C4 pad. Thus, by including additional FSOURCE IO fatwire cells in the design of a fuse domain, it is possible to place primary fuse macros at convenient locations on the semiconductor chip. The fuse controller can

control fuse programming over a normal (e.g. not a fatwire) connector since control currents are small.

[Para 39] In a preferred embodiment, the present invention will be implemented in fuse insertion software. The software is modified to break up the fuses in the fuse domain into a user-specified number of subgroups. Each subgroup of fuses is connected to its own FSOURCE fatwire IO. A threshold will be used to determine the maximum number of fuse macros per subgroup during fuse insertion. This threshold will be set to a default for each new design and can be changed based on learning during floorplanning for the next iteration of the design.

[Para 40] A software algorithm for determining the maximum number of fuse macros per subgroup assumes that a fuse domain contains the following components:

[Para 41] A single fuse controller macro

[Para 42] A single FSOURCE C4 pad

[Para 43] A single fatwire IO cell

[Para 44] N number of primary fuse macros

[Para 45] Fuse domain overhead (secondary fuses, tertiary fuses, etc...)

[Para 46] The general algorithm for software used in chip floorplanning is illustrated in Figure 6, and is described in detail as follows:

[Para 47] 1. Determine P and N.

[Para 48] Determine the number of primary fuse macros P required for the design based on the amount RAM present in the design. Those of skill in the art will recognize that the number of primary fuse macros typically increases as the amount of RAM increases. Let N be the maximum allowable primary fuse macros per FSOURCE connection. N is determined by experience or rule of thumb, as some number less than the maximum number of primary fuse

macros allowable in a domain that can be successfully floorplanned on an average design. This means that on an average design, an FSOURCE fatwire IO may connect to up to N primary fuse macros without overly perturbing the floorplan of critical logic or without causing any fuse macro to become excessively distal from its associated FSOURCE connection.

[Para 49]   2. Compare P and N and create fuse macros.

[Para 50]   In general, if the number of primary fuse macros, P, required for the design is less than N (P<N), then the software creates one FSOURCE C4. However, if the number of primary fuse macros required by the design is greater than N (P>N), additional FSOURCE C4s are created by the software. The total number of FSOURCE C4s created initially is P/N, rounded up to the nearest whole number.

[Para 51]   3. Divide primary fuse macros among FSOURCE C4s.

[Para 52]   The primary fuse macros are divided among the FSOURCE C4s that are created initially. Division of the primary fuse macros can be done in any suitable manner. For example, the division may either be done randomly or in the order in which the fuse macros are inserted into the netlist. Those of skill in the art will recognize that the method of division is not critical since the affinity of the fuse macros is to the FSOURCE C4 itself, not to each other. Also, it is possible later to reconnect fuse macros to the nearest FSOURCE based on placement as long the FSOURCE is within the same domain. In a preferred embodiment, the primary fuse macros are divided evenly among the FSOURCE C4s and fatwire IO cells, or as near evenly as possible if even division is not possible (e.g. if there is an odd number of fuse macros).

[Para 53]   4. Floorplan the chip.

[Para 54]   If the normal floorplanning constraints (timing, IO wiring, etc.) cannot be met due to the FSOURCE constraints, go on to step 5.

[Para 55]   5. Increase the number of FSOURCE C4s by 1.

[Para 56]   6. Divide the primary fuse macros among the FSOURCE C4s (as described in step 1).

[Para 57]   7. Repeat steps 4–6 until normal floorplanning constraints are met in step 4.

[Para 58]   Repetition of steps 4–6 may be carried out as many times as is necessary until normal floorplanning constraints are met.

[Para 59]   For example, according to a floorplanning software system, the default number of primary fuse macros per FSOURCE C4 in a floorplan is 32 (i.e. N = 32). If the number of primary fuse macros (P) required in a floorplan is less than 32, one FSOURCE C4 is created by the software. However, if the number of primary fuse macros P required in a floorplan is greater than 32, P/N (rounded up to the nearest whole number) FSOURCE C4s are created by the software and the fuse macros are divided among them. Floorplanning is then undertaken.

[Para 60]   If it is discovered through floorplanning that even more FSOURCE C4s are required in order to physically spread out the fuse macros sufficiently, the number of FSOURCE C4s for the domain is increased, the number of primary fuse macros per FSOURCE is decreased, and the software is rerun.

[Para 61]   Another benefit of the fuse domain design of the present invention is that fuses inside a single domain but connected to different FSOURCEs can be blown in parallel, by suitable fuse controller design, thus saving time during the fuse blow during manufacturing tests.

[Para 62]   The foregoing examples serve to illustrate the practice of the present invention but should not be interpreted so as to limit the invention in any way.

[Para 63]   *EXAMPLES*

[Para 64]   EXAMPLE 1.

[Para 65] 1) A fuse domain has 28 primary fuses, i.e. P = 28. The default number of primary fuses per primary FSOURCE is 32, i.e. N = 32 in the software system. The ratio of primary fuses to default primary fuses (P/N) is thus 28 / 32. Rounding the fraction up to the nearest whole number gives a value of 1, so this fuse domain receives 1 primary FSOURCE C4. The Fuse Macros are clustered around this single FSOURCE C4.

[Para 66] 2) If this fuse domain grows to 35 primary fuses, then the P/N ratio becomes 35 / 32. Again, rounding this fraction up to the nearest whole number this time gives a value of 2. This expanded fuse domain receives two primary FSOURCE C4s. The fuse macros are clustered around their associated primary FSOURCE C4 on the chip.

[Para 67] 3) Through floorplanning, it is determined that this domain requires an additional supplemental FSOURCE C4, or a total of 3 FSOURCE C4s. The default number of primary fuses per FSOURCE (N) is set to 15 as an input to the software. Setting N to 15 increases the number of FSOURCEs whereas changing N to 20 yields only 2 FSOURCEs. The choice of 15 is somewhat arbitrary – the desired effect is to yield 3 FSOURCEs out of the equation. The P/N ratio becomes 35/15, which when rounded up to the nearest whole number gives a value of 3. The fuse domain is assigned 3 FSOURCE C4s. The Fuse Macros are clustered around their associated FSOURCE C4 on the chip.

[Para 68] While the invention has been described in terms of its preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims. Accordingly, the present invention should not be limited to the embodiments as described above, but should further include all modifications and equivalents thereof within the spirit and scope of the description provided herein.